

Digital Ecosystems: Evolving Service-Oriented Architectures

Gerard Briscoe

Intelligent Systems and Networks Group
Department of Electrical and Electronic Engineering
Imperial College London
London, United Kingdom, SW7 2BT
e-mail: gerard.briscoe@ic.ac.uk

Philippe De Wilde

Intelligent Systems Lab
School of Mathematical and Computer Science
Heriot-Watt University
Edinburgh, United Kingdom, EH14 4AS
e-mail: pdw@macs.hw.ac.uk

Abstract— We view Digital Ecosystems to be the *digital counterparts of biological ecosystems*, exploiting the self-organising properties of biological ecosystems, which are considered to be robust, self-organising and scalable architectures that can automatically solve complex, dynamic problems. Digital Ecosystems are a novel optimisation technique where the optimisation works at two levels: a first optimisation, migration of agents (representing services) which are distributed in a decentralised peer-to-peer network, operating continuously in time; this process feeds a second optimisation based on evolutionary computing that operates locally on single peers and is aimed at finding solutions to satisfy locally relevant constraints. We created an Ecosystem-Oriented Architecture (EOA) of Digital Ecosystems by extending Service-Oriented Architectures (SOAs) with distributed evolutionary computing (DEC), allowing services to recombine and evolve over time, constantly seeking to improve their effectiveness for the user base. Individuals within our Digital Ecosystem will be applications (groups of services), created in response to user requests by using evolutionary optimisation to aggregate the services. These individuals will migrate through the Digital Ecosystem and adapt to find *niches* where they are useful in fulfilling other user requests for applications. Simulation results imply that the Digital Ecosystem performs better at large scales than a comparable SOA, suggesting that incorporating ideas from theoretical ecology can contribute to useful self-organising properties in digital ecosystems.

I. INTRODUCTION

Is mimicking ecosystems the future of information systems? A key challenge in modern computing is to develop systems that address complex, dynamic problems in a scalable and efficient way, because the increasing complexity of software makes designing and maintaining efficient and flexible systems a growing challenge [2], [3], [4]. What with the ever expanding number of services being offered online from Application Programming Interfaces (APIs) being made public, there is an ever growing number of computational units available to be combined in the creation of applications. However, this is currently a task done manually by programmers, and it has been argued that current software development techniques have hit a *complexity wall* [5], which can only be overcome by automating the search for new algorithms. There are several existing efforts aimed at achieving this automated service composition [6], [7], [8], [9], the most prevalent of which is Service-Oriented Architectures and its associated standards and technologies [10], [11].

Alternatively, nature has been in the research business for 3.8 billion years and in that time has accumulated close to

30 million *well-adjusted* solutions to a plethora of design challenges that humankind struggles to address with mixed results [12]. Biomimicry is a discipline that seeks solutions by emulating nature's designs and processes, and there is considerable opportunity to learn elegant solutions for human-made problems [12]. Biological ecosystems are thought to be robust, scalable architectures that can automatically solve complex, dynamic problems, possessing several properties that may be useful in automated systems. These properties include self-organisation, self-management, scalability, the ability to provide complex solutions, and automated composition of these complex solutions [13].

Therefore, an approach to the aforementioned challenge would be to develop Digital Ecosystems, artificial systems that aim to harness the dynamics that underlie the complex and diverse adaptations of living organisms in biological ecosystems. While evolution may be well understood in computer science under the auspices of *evolutionary computing* [14], ecological models are not. The possible connections between Digital Ecosystems and their biological counterparts are yet to be closely examined, so potential exists to create an Ecosystem-Oriented Architecture with the essential elements of biological ecosystems, where the word *ecosystem* is more than just a *metaphor*. We propose that an ecosystem inspired approach, would be more effective at greater scales than traditionally inspired approaches, because it would be built upon the scalable and self-organising properties of biological ecosystems [13].

II. SERVICE-ORIENTED ARCHITECTURES

Our approach to evolving high-level software applications requires a modular reusable paradigm to software development. Service-oriented architectures (SOAs) are the current state-of-the-art approach, being the current iteration of interface/component-based design from the 1990s, which was itself an iteration of event-oriented design from the 1980s, and before then modular programming from the 1970s [15], [16]. Service-oriented computing promotes assembling application components into a loosely coupled network of services, to create flexible, dynamic business processes and agile applications that span organisations and computing platforms [17]. This is achieved through a SOA, an architectural style that guides all aspects of creating and using business processes throughout their life-cycle, packaged as services. This includes defining and provisioning the infrastructure that allows different applications to exchange data and participate in business processes,

loosely coupled from the operating systems and programming languages underlying the applications [18]. Hence, a SOA represents a model in which functionality is decomposed into distinct units (services), which can be distributed over a network, and can be combined and reused to create business applications [17].

A SOA depends upon service-orientation as its fundamental design principle. In a SOA environment, independent services can be accessed without knowledge of their underlying platform implementation [18]. Services reflect a *service-oriented* approach to programming that is based on composing applications by discovering and invoking network-available services to accomplish some task. This approach is independent of specific programming languages or operating systems, because the services communicate with each other by passing data from one service to another, or by co-ordinating an activity between two or more services [17]. So, the concepts of SOAs are often seen as built upon, and the development of, the concepts of modular programming and distributed computing [16].

SOAs allow for an information system architecture that enables the creation of applications that are built by combining loosely coupled and interoperable services [18]. They typically implement functionality most people would recognise as a service, such as filling out an online application for an account, or viewing an online bank statement [16]. Services are intrinsically un-associated units of functionality, without calls to each other embedded in them. Instead of services embedding calls to each other in their source code, protocols are defined which describe how services can talk to each other, in a process known as orchestration, to meet new or existing business system requirements [19]. This is allowing an increasing number of third-party software companies to offer software services, such that SOA systems will come to consist of such third-party services combined with others created in-house, which has the potential to spread costs over many users and uses, and promote standardisation both in and across industries [20]. For example, the travel industry now has a well-defined and documented, set of both services and data, sufficient to allow any competent software engineer to create travel agency software using entirely off-the-shelf software services [21], [22]. Other industries, such as the finance industry, are also making significant progress in this direction [23].

The vision of SOAs assembling application components from a loosely coupled network of services that can create dynamic business processes and agile applications that span organisations and computing platforms, is visualised in Figure 1. It will be made possible by creating compound solutions that use internal organisational software assets, including enterprise information and legacy systems, and combining these solutions with external components residing in remote networks [24]. The great promise of SOAs is that the *marginal cost* of creating the *n*-th application is virtually zero, as all the software required already exists to satisfy the requirements of other applications. Only their *combination* and *orchestration* are required to produce a new application [25], [26]. The *key* is that the interactions between the *chunks*, are not specified within the *chunks* themselves. Instead, the interaction of services (all of whom are hosted by un-associated peers) is specified by users in an ad-hoc way, with the intent driven by

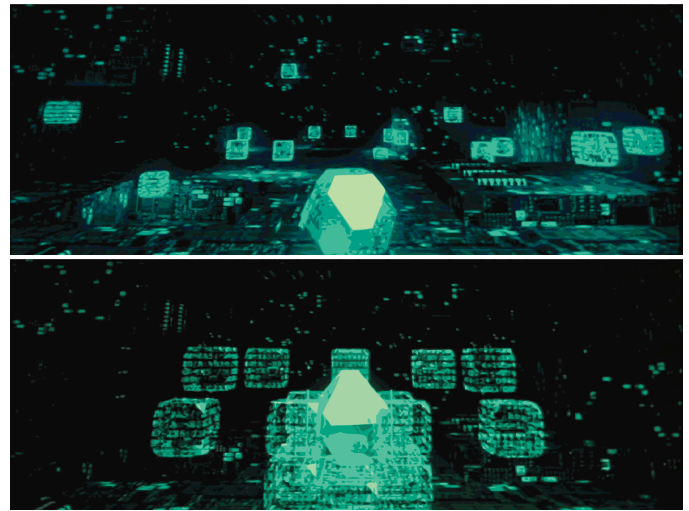


Fig. 1. *Service-Oriented Architectures: Abstract visualisations, with the first image showing the loosely joined services as cuboids, and the service orchestration as a polyhedron; and the second image showing their high interoperability and re-usability in forming applications, from the use of standardised interfaces and external service orchestration.*

newly emergent business requirements [27].

The pinnacle of SOA interoperability, is the exposing of services on the internet as *web services* [18]. A web service is a specific type of service that is identified by a Uniform Resource Identifier (URI), whose service description and transport utilise open Internet standards. Interactions between web services typically occur as Simple Object Access Protocol (SOAP) calls carrying eXtensible Markup Language (XML) data content. Interface descriptions of the web services are expressed using the Web Services Definition Language (WSDL) [28]. The Universal Description Discovery and Integration (UDDI) standard defines a protocol for directory services that contain web service descriptions. UDDI enables web service clients to locate candidate services and discover their details. Service clients and service providers utilise these standards to perform the basic operations of SOAs [28]. Service aggregators can then use the Business Process Execution Language (BPEL) to create new web services by defining corresponding compositions of the interfaces and internal processes of existing services [28].

SOA services inter-operate based on a formal definition (or contract, e.g. WSDL) that is independent of the underlying platform and programming language. Service descriptions are used to advertise the service capabilities, interface, behaviour, and quality [28]. The publication of such information about available services provides the necessary means for discovery, selection, binding, and composition of services [28]. The (expected) behaviour of a service during its execution is described by its behavioural description (for example, as a workflow process). Also, included is a quality of service (QoS) description, which publishes important functional and non-functional service quality attributes, such as service metering and cost, performance metrics (response time, for instance), security attributes, integrity (transactional), reliability, scalability, and availability [28]. Service clients (end-user organisations that use some service) and service aggregators (organisations that consolidate multiple services into a new, single service offering) utilise *service descriptions* to achieve their objectives [28]. One of the most important and continuing developments in SOAs is the use of *semantic*

descriptions for service discovery, so that a client can discover the services semantically, and then apply transformations to adapt the interface of the services to the interface expected, using already available client software [29].

There are multiple standards available and still being developed for SOAs [11], most notably of recent being REpresentational State Transfer (REST) [19]. The software industry now widely implements a thin SOAP/WSDL/UDDI veneer atop existing applications or components that implement the web services paradigm [24], but the choice of technologies could change with time. Therefore, SOAs and its services are best defined generically, because SOAs are technology agnostic and need not be tied to a specific technology [17]. Within the current and future scope of SOAs, there is clearly potential to *evolve* complex high-level software applications from the modular services of SOAs, instead of the instruction level evolution currently prevalent in genetic programming [30].

III. DISTRIBUTED EVOLUTIONARY COMPUTING

The fact that evolutionary computing manipulates a population of independent solutions actually makes it well suited for parallel computation architectures [31]. The motivation for using parallel or distributed evolutionary algorithms is twofold. First, improving the speed of evolutionary processes by conducting concurrent evaluations of individuals in a population. Second, improving the problem-solving process by overcoming difficulties that face traditional evolutionary algorithms, such as maintaining diversity to avoid premature convergence [32], [33]. There are several variants of distributed evolutionary computing, leading some to propose a taxonomy for their classification [34], with there being two main forms of models [31], [33]:

- multiple-population/coarse-grained migration/island
- single-population/fine-grained diffusion/neighbourhood

In the coarse-grained *island* models [35], [31], evolution occurs in multiple parallel sub-populations (islands), each running a local evolutionary algorithm, evolving independently with occasional *migrations* of highly fit individuals among sub-populations. The core parameters for the evolutionary algorithm of the island-models are as follows [35]:

- number of the sub-populations: 2, 3, 4, more
- sub-population homogeneity
 - size, crossover rate, mutation rate, migration interval
- topology of connectivity: ring, star, fully-connect, random
- static or dynamic connectivity
- migration mechanisms:
 - isolated/synchronous/asynchronous
 - how often migrations occur
 - which individuals migrate

Fine-grained *diffusion* models [36], [33] assign one individual per processor. A local neighbourhood topology is assumed, and individuals are allowed to mate only within their neighbourhood, called a *deme*. The demes overlap by an amount that depends on their shape and size, and in this way create an implicit migration mechanism. Each processor runs an identical evolutionary algorithm which selects parents from the local neighbourhood, produces an offspring, and decides whether to replace the current individual with an offspring. However, even with the advent of multi-processor

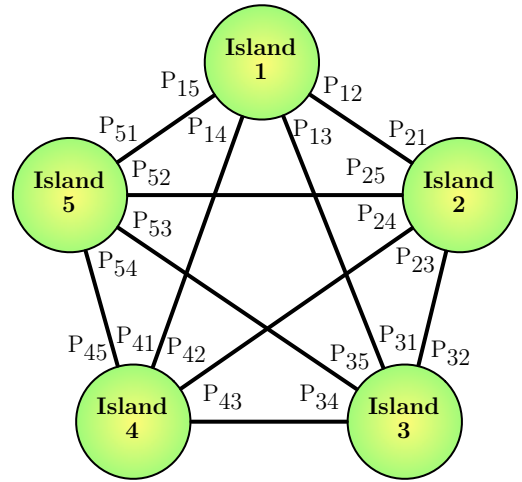


Fig. 2. Island-Model of Distributed Evolutionary Computing [35], [31]: There are different probabilities of going from island ① to island ②, as there is of going from island ② to island ①. This mirrors the naturally inspired quality that although two populations have the same physical separation, it may be easier to migrate in one direction than the other, i.e. fish migration is easier downstream than upstream.

computers, and more recently multi-core processors, which provide the ability to execute multiple threads simultaneously [4], this approach would still prove impractical in supporting the number of agents necessary to create a Digital Ecosystem. Therefore, we shall further consider the *island* models.

An example island-model [35], [31] is visualised in Figure 2, in which there are different probabilities of going from island ① to island ②, as there is of going from island ② to island ①. This allows maximum flexibility for the migration process, and mirrors the naturally inspired quality that although two populations have the same physical separation, it may be easier to migrate in one direction than the other, i.e. fish migration is easier downstream than upstream. The migration of the *island* models is like the notion of migration in nature, being similar to the metapopulation models of theoretical ecology [37]. This model has also been used successfully in the determination of investment strategies in the commercial sector, in a product known as the Galapagos toolkit [38], [39]. However, all the *islands* in this approach work on exactly the same problem, which makes it less analogous to biological ecosystems in which different locations can be environmentally different [40]. We will take advantage of this property later when defining the Ecosystem-Oriented Architecture of Digital Ecosystems.

IV. THE DIGITAL ECOSYSTEM

We are concerned with the digital counterpart of biological ecosystems. However, the term *digital ecosystem* has been used to describe a variety of concepts, which it now makes sense to review. Some of these refer to the existing networking infrastructure of the internet [41], [42], [43], while several companies offer a *digital ecosystem* service or solution, which involves enabling customers to use existing e-business solutions [44], [45], [46]. The term is also being increasingly linked, yet undefined, to the future developments of Information and Communications Technology (ICT) adoption for e-business and e-commerce, to create so called *business ecosystems* [47], [48], [49]. However, perhaps the most frequent references to *digital ecosystems* arise in Artificial Life research, where they are created primarily to

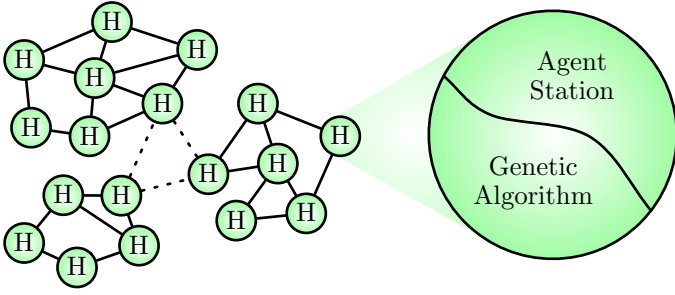


Fig. 3. *Digital Ecosystem: Optimisation architecture in which agents (representing services) travel along the peer-to-peer connections; in every node (habitat) local optimisation is performed through an evolutionary algorithm, where the search space is determined by the agents present at the node.*

investigate aspects of biological and other complex systems [50], [51], [52]. The extent to which these disparate systems resemble biological ecosystems varies, and frequently the word *ecosystem* is merely used for branding purposes without any inherent ecological properties.

We consider Digital Ecosystems [53], [54], [55] to be software systems that exploit the properties of biological ecosystems, which are robust, scalable, and self-organising [13]. So, Digital Ecosystems provide a two-level optimisation scheme inspired by natural ecosystems, in which a decentralised peer-to-peer network forms an underlying tier of distributed agents. These agents then feed a second optimisation level based on an evolutionary algorithm that operates locally on single habitats (peers), aiming to find solutions that satisfy locally relevant constraints. The local search is sped up through this twofold process, providing better local optima as the distributed optimisation provides prior sampling of the search space by making use of computations already performed in other peers with similar constraints [53]. The agents consist of an *executable component* and an *ontological description* [56]. So, the Digital Ecosystem can be considered a Multi-Agent System (MAS) [56] which uses *distributed evolutionary computing* [31], [33] to combine suitable agents in order to meet user requests for applications.

The landscape, in energy-centric biological ecosystems, defines the connectivity between habitats [40]. Connectivity of nodes in the digital world is generally not defined by geography or spatial proximity, but by information or semantic proximity. For example, connectivity in a peer-to-peer network is based primarily on bandwidth and information content, and not geography. The island-models of distributed evolutionary computing use an information-centric model for the connectivity of nodes (*islands*) [35]. However, because it is generally defined for one-time use (to evolve a solution to one problem and then stop) it usually has a fixed connectivity between the nodes, and therefore a fixed topology [31]. So, supporting evolution in the Digital Ecosystem, with a multi-objective *selection pressure* (fitness landscape [57] with many peaks), requires a re-configurable network topology, such that habitat connectivity can be dynamically adapted based on the observed migration paths of the agents between the users within the habitat network. Based on the island-models of distributed evolutionary computing [35], each connection between the habitats is bi-directional and there is a probability associated with moving in either direction across the connection, with the connection probabilities affecting the rate of migration of the agents. However, additionally, the connection

probabilities will be updated by the success or failure of agent migration using the concept of Hebbian learning [58]: the habitats which do not successfully exchange agents will become less strongly connected, and the habitats which do successfully exchange agents will achieve stronger connections. This leads to a topology that adapts over time, resulting in a network that supports and resembles the connectivity of the user base. If we consider a *business ecosystem*, network of Small and Medium sized Enterprises, as an example user base; such business networks are typically small-world networks [59], [60]. They many strongly connected clusters (communities), called *sub-networks* (quasi-complete graphs), with a few connections between these clusters (communities) [61]. Graphs with this topology have a very high clustering coefficient and small characteristic path lengths [61]. So, the Digital Ecosystem will take on a topology similar to that of the user base.

The novelty of our approach comes from the evolving populations being created in response to *similar* requests. So whereas in the island-models of distributed evolutionary computing there are multiple evolving populations in response to one request [35], here there are multiple evolving populations in response to *similar* requests. In our Digital Ecosystems different requests are evaluated on separate *islands* (populations), and so adaptation is accelerated by the sharing of solutions between evolving populations (islands), because they are working to solve similar requests (problems).

The users will formulate queries to the Digital Ecosystem by creating a request as a *semantic description*, like those being used and developed in SOAs [29], specifying an application they desire and submitting it to their local peer (habitat). This description defines a metric for evaluating the *fitness* of a composition of agents, as a distance function between the *semantic description* of the request and the agents' *ontological descriptions*. A population is then instantiated in the user's habitat in response to the user's request, seeded from the agents available at their habitat. This allows the evolutionary optimisation to be accelerated in the following three ways: first, the habitat network provides a subset of the agents available globally, which is localised to the specific user it represents; second, making use of agent-sequences previously evolved in response to the user's earlier requests; and third, taking advantage of relevant agent-sequences evolved elsewhere in response to similar requests by other users. The population then proceeds to evolve the optimal agent-sequence(s) that fulfils the user request, and as the agents are the base unit for evolution, it searches the available agent-sequence combination space. For an evolved agent-sequence that is executed (instantiated) by the user, it then migrates to other peers (habitats) becoming hosted where it is useful, to combine with other agents in other populations to assist in responding to other user requests for applications.

V. SIMULATION AND RESULTS

An important measure for determining the success of the Digital Ecosystem is its relative performance to a traditional SOA based system. So, we simulated a simple SOA with a distributed UDDI service registry [17], with redirects (links to other nodes) at each node for service descriptions not stored locally, and without caching (analogous to the Digital Ecosystem). We then compared it against a typical simulation

run of the Digital Ecosystem. The time available to the SOA based system for fulfilling a request from the distributed registry was limited to the time the Digital Ecosystem required to respond to the same user request, and worked by searching for the optimal services based on the optimal segmentation of the request, because an exhaustive combinatorial search would have been impractical. We simulated the Digital Ecosystem, following the Ecosystem-Oriented Architecture from the previous section. Throughout the simulations we assumed a hundred users, which meant that at any time the number of users joining the network equalled those leaving. The habitats of the users were randomly connected at the start, to simulate the users going online for the first time. The users then produced agents (services) and requests for business applications. Initially, the users each deployed five agents to their habitats, for migration (distribution) to any habitats connected to theirs (i.e. their community). Users were simulated to deploy a new agent after the submission of three requests for business applications, and were chosen at random to submit their requests.

A simulated user request consisted of an abstract *semantic description*, as a list of sets of numeric tuples to represent the properties of a desired business application. The use of the *numeric tuples* made it comparable to the *semantic descriptions* of the services represented by the agents; while the *list of sets* (two level hierarchy) and a much longer length provided sufficient complexity to support the sophistication of business applications.

In the Digital Ecosystem user requests were handled by the habitats instantiating evolving populations, which used evolutionary computing to find the optimal solution(s), agent-sequence(s). It was assumed that the users made their requests for business applications *accurately*, and always used the response (agent-sequence) provided. Populations of agents, $[A_1, A_1, A_2, \dots]$, were evolved to solve user requests, seeded with agents and agent-sequences from the *agent-pool* of the habitats in which they were instantiated. A dynamic population size was used to ensure exploration of the available combinatorial search space, which increased with the average length of the population's agent-sequences. The optimal combination of agents (agent-sequence) was evolved to the user request, R , by an artificial *selection pressure* created by a *fitness function* generated from the user request, R . An individual (agent-sequence) of the population consisted of a set of attributes, a_1, a_2, \dots , and a user request essentially consisted of a set of required attributes varied according to a Gaussian distribution, r_1, r_2, \dots . So, the *fitness function* for evaluating an individual agent-sequence, A , relative to a user request, R , was,

$$fitness(A, R) = \frac{1}{1 + \sum_{r \in R} |r - a|}, \quad (1)$$

where a is the member of A such that the difference to the required attribute r was minimised. Equation 1 was used to assign *fitness* values between 0.0 and 1.0 to each individual of the current generation of the population, directly affecting their ability to replicate into the next generation. The evolutionary computing process was encoded with a

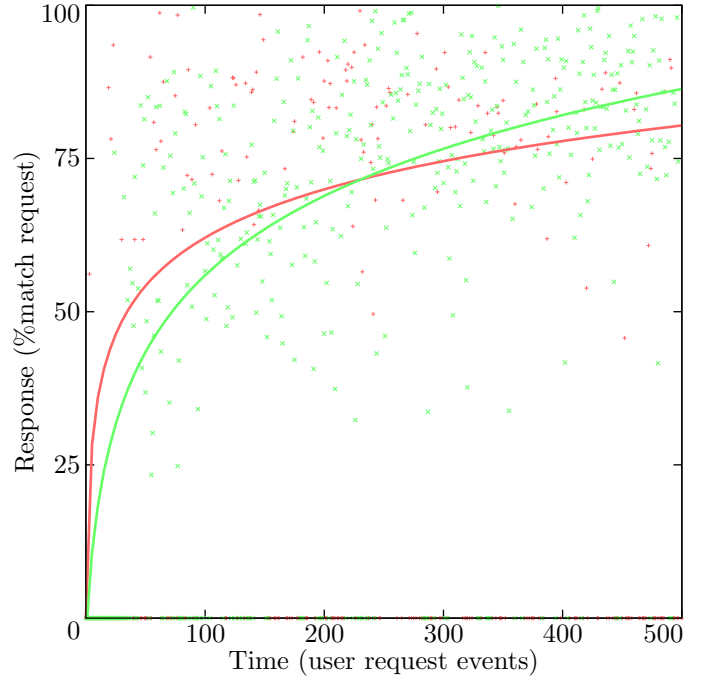


Fig. 4. Graph of the performance of the Digital Ecosystem against a traditional SOA based system: Both the Digital Ecosystem and the SOA based system performed as expected, providing better responses to user requests as more services became available. The SOA reference system initially performed better than the Digital Ecosystem, but with the increasing number of services the Digital Ecosystem outperformed the reference system.

low mutation rate, a fixed selection pressure and a non-trapping fitness function (i.e. did not get trapped at local optima). The type of selection used *fitness-proportional* and *non-elitist*. *Fitness-proportional* meaning that the fitter the individual the higher its probability of surviving to the next generation [62]. *Non-elitist* meaning that the best individual from one generation was not guaranteed to survive to the next generation; it had a high probability of surviving into the next generation, but it was not guaranteed as it might have been mutated, [14]. *Crossover* (recombination) was then applied to a randomly chosen 10% of the surviving population, a *one-point crossover*, by aligning two parent individuals and picking a random point along their length, and at that point exchanging their tails to create two offspring [14]. *Mutations* were then applied to a randomly chosen 10% of the surviving population; *point mutations* were randomly located, consisting of *insertions* (an agent was inserted into an agent-sequence), *replacements* (an agent was replaced in an agent-sequence), and *deletions* (an agent was deleted from an agent-sequence) [63]. The issue of bloat was controlled by augmenting the *fitness function* with a *parsimony pressure* [64] which biased the search to shorter agent-sequences, evaluating longer than average length agent-sequences with a reduced *fitness*, and thereby providing a dynamic control limit which adapted to the average length of the ever-changing evolving agent populations.

In Figure 4 we graphed the percentage match to the user requests for typical runs, as determined by a distance function between the request and the service descriptions. Both the Digital Ecosystem and the SOA based system performed as expected, providing better responses to user requests as more services became available. The SOA reference system initially performed better than the Digital Ecosystem, but with the increasing number of services the Digital Ecosystem

outperformed the reference system. This was anticipated as the the Digital Ecosystem was expected to be more effective at larger scales [53].

VI. CONCLUSION

The experimental results indicate that under simulation conditions the Digital Ecosystem outperforms the comparison system based on a traditional Service-Oriented Architecture. Service-oriented architectures promise to provide potentially huge numbers of services that programmers can combine via standardised interfaces, to create increasingly sophisticated and distributed applications [28]. The Digital Ecosystem extends this concept with the automatic combining of available and applicable services in a scalable architecture to meet user requests for applications. This is made possible by a fundamental paradigm shift, from a *pull*-oriented approach to a *push*-oriented approach. So, instead of the *pull*-oriented approach of generating applications only upon request in Service-Oriented Architectures [19], the Digital Ecosystem follows a *push*-oriented approach of distributing and composing applications pre-emptively, as well as upon request. Although the use of Service-Oriented Architectures in the definition of Digital Ecosystems provides a predisposition to business [16], it does not preclude other more general uses. The Ecosystem-Oriented Architecture definition of Digital Ecosystems is intended to be inclusive and interoperable with other technologies, in the same way that the definition of Service-Oriented Architectures is with *grid computing* and other technologies [19]. For example, habitats could be executed using a distributed processing arrangement, such as *cloud computing* [65], which would be possible because the habitat network topology is information-centric (instead of location-centric).

ACKNOWLEDGMENT

The authors would like to thank for their encouragement and suggestions: Paolo Dini of the London School of Economics and Political Science, Thomas Heistracher and his group of the Salzburg Technical University, Jonathan Rowe of the University of Birmingham, and Miguel Vidal of Sun Microsystems. The Digital Ecosystem model was constructed through interacting with these people and others.

REFERENCES

- [1] A. English. Digital Business Ecosystems (DBE). [Online]. Available: http://www.digital-ecosystem.org/DBE_Main/about
- [2] S. Miller, "Aspect-oriented programming takes aim at software complexity," *IEEE Computer*, vol. 34, pp. 18–21, 2001.
- [3] H. Sutter, "The free lunch is over: A fundamental turn toward concurrency in software," *Dr. Dobbs's Journal*, 2005, cr newsArticle2.
- [4] J. Markoff, "Faster chips are leaving programmers in their dust," *New York Times*, Tech. Rep., 2007. [Online]. Available: <http://www.nytimes.com/2007/12/17/technology/17chip.html>
- [5] K. Lyytinen and Y. Yoo, "The next wave of nomadic computing: A research agenda for information systems research," *Sprouts: Working Papers on Information Systems*, vol. 1, pp. 1–20, 2001.
- [6] S. McIlraith, C. Son, and H. Zeng, "Semantic web services," *IEEE Intelligent Systems*, vol. 16, pp. 46–53, 2001.
- [7] S. Narayanan and S. McIlraith, "Simulation, verification and automated composition of web services," in *international conference on World Wide Web*. ACM Press, 2002, pp. 77–88.
- [8] N. Milanovic and M. Malek, "Current solutions for web service composition," *IEEE Internet Computing*, vol. 8, pp. 51–59, 2004.
- [9] J. Rao and X. Su, "A survey of automated web service composition methods," in *Semantic Web Services and Web Process Composition*, J. Cardoso and A. Sheth, Eds. Springer, 2004, pp. 43–54.
- [10] F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, and S. Weerawarana, "Unraveling the web services web: An introduction to SOAP, WSDL, and UDDI," *IEEE Internet Computing*, vol. 6, pp. 86–93, 2002.
- [11] B. Violino. (2007) How to navigate a sea of SOA standards. [Online]. Available: http://www.cio.com/article/104007/How_to_Navigate_a_Sea_of_SOA_Standards
- [12] J. Benyus, *Biomimicry, Innovation Inspired by Nature*. Harper Collins Publishers, 2002.
- [13] S. Levin, "Ecosystems and the biosphere as complex adaptive systems," *Ecosystems*, vol. 1, pp. 431–436, 1998.
- [14] A. Eiben and J. Smith, *Introduction to Evolutionary Computing*. Springer, 2003.
- [15] E. Canal, "Using the ESB with e4: Applying new tendencies to CSC business integration architecture," Computer Sciences Corporation, Tech. Rep., 2007. [Online]. Available: <http://www.csc.com/aboutus/leadingedgeforum/knowledgelibrary/uploads/Using%20the%20ESB%20with%20e4%20-%20ERCanal.pdf>
- [16] D. Krafzig, K. Banke, and D. Slama, *Enterprise SOA: Service-Oriented Architecture Best Practices*. Prentice Hall, 2004.
- [17] M. Papazoglou, "Service-oriented computing: concepts, characteristics and directions," in *International Conference on Web Information Systems Engineering*, T. Catarci, M. Mecella, J. Mylopoulos, and M. Orłowska, Eds. IEEE Press, 2003, pp. 3–12.
- [18] E. Newcomer and G. Lomow, *Understanding SOA with web services*. Addison-Wesley, 2005.
- [19] M. Singh and M. Huhns, *Service-Oriented Computing: Semantics, Processes, Agents*. Wiley, 2005.
- [20] A. Chhatpar, "Increase business agility through BRM systems and SOA: Best bets to increase ROI on your enterprise applications," IBM Developer Works, Tech. Rep., 2008. [Online]. Available: <http://download.boulder.ibm.com/ibmdl/pub/software/dw/architecture/ar-brmssoa/ar-brmssoa-pdf.pdf>
- [21] A. Kotok, *ebXML: The New Global Standard for Doing Business Over the Internet*. Sams Publishing, 2001.
- [22] J. Cardoso and A. Sheth, "Introduction to semantic web services and web process composition," in *Semantic Web Services and Web Process Composition*, J. Cardoso and A. Sheth, Eds. Springer, 2004, pp. 1–13.
- [23] O. Zimmermann, S. Milinski, M. Craes, and F. Oellermann, "Second generation web services-oriented architecture in production in the finance industry," in *Conference on Object-Oriented Programming, Systems, Languages, and Applications*, J. Vlissides and D. Schmidt, Eds. ACM Press, 2004, pp. 283–289.
- [24] M. Papazoglou, P. Traverso, S. Dustdar, and F. Leymann, "Service-oriented computing: State of the art and research challenges," *IEEE Computer*, vol. 40, pp. 38–45, 2007.
- [25] Q. Tang, "Economics of web service provisioning: Optimal market structure and intermediary strategies," Ph.D. dissertation, University of Florida, 2004.
- [26] G. Modi, "Service oriented architecture & web 2.0," Guru Tegh Bahadur Institute of Technology, Tech. Rep., 2007. [Online]. Available: http://www.gsmodi.com/files/SOA_Web2_Report.pdf
- [27] F. Leymann, D. Roller, and M. Schmidt, "Web services and business process management," *IBM Systems Journal*, vol. 41, pp. 198–211, 2002.
- [28] M. Papazoglou and D. Georgakopoulos, "Service-oriented computing," *Communications of the ACM*, vol. 46, pp. 25–28, 2003, cr 4.5.
- [29] P. Rajasekaran, J. Miller, K. Verma, and A. Sheth, "Enhancing web services description and discovery to facilitate composition," in *Semantic Web Services and Web Process Composition*, J. Cardoso and A. Sheth, Eds. Springer, 2004, pp. 55–68.
- [30] J. Koza, "Overview of genetic programming," in *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992, pp. 73–78, cr koza:92.
- [31] E. Cantu-Paz, "A survey of parallel genetic algorithms," *Réseaux et systèmes répartis, Calculateurs Parallèles*, vol. 10, pp. 141–171, 1998.
- [32] H. Muhlenbein, "Evolution in time and space - the parallel genetic algorithm," *Foundations of Genetic Algorithms*, vol. 1, pp. 316–337, 1991.
- [33] J. Stender, *Parallel Genetic Algorithms: Theory and Applications*. IOS Press, 1993.
- [34] M. Nowostawski and R. Poli, "Parallel genetic algorithm taxonomy," in *International Conference on Knowledge-Based Intelligent Information Engineering Systems*, L. Jain, Ed. IEEE Press, 1999, pp. 88–92.
- [35] S. Lin, W. Punch III, and E. Goodman, "Coarse-grain parallel genetic algorithms: categorization and new approach," in *Symposium on Parallel and Distributed Processing*. IEEE Press, 1994, pp. 28–37.
- [36] B. Manderick and P. Spiessens, "Fine-grained parallel genetic algorithms," in *International Conference on Genetic Algorithms*, J. Schaffer, Ed. Morgan Kaufmann Publishers, 1989, pp. 428–433.

- [37] R. Levins, "Some demographic and genetic consequences of environmental heterogeneity for biological control," *Bulletin of the Entomological Society of America*, vol. 15, pp. 237–240, 1969.
- [38] M. Ward. (2004) Life offers lessons for business. [Online]. Available: <http://news.bbc.co.uk/1/hi/technology/3752725.stm>
- [39] Codefarm Software Limited. (2008) Codefarm - technology for structured credit. [Online]. Available: <http://www.codefarm.com/>
- [40] M. Begon, J. Harper, and C. Townsend, *Ecology: Individuals, Populations and Communities*. Blackwell Publishing, 1996.
- [41] P. Denning and R. Metcalfe, *Beyond Calculation: The Next Fifty Years of Computing*. Springer, 1997.
- [42] C. Fiorina. (2000) The digital ecosystem. [Online]. Available: http://www.hp.com/hpinfo/execteam/speeches/fiorina/ceo_worldres.00.html
- [43] Ximbiotix. (2005) About the digital ecosystem. [Online]. Available: <http://www.ximbiotix.com/desktop/the-digital-ecosystem/about-the-digital-ecosystem.cfm>
- [44] D. Bennett, "Digital transformation in the entertainment industry - embracing the fully digital ecosystem," Accenture, Tech. Rep., 2006. [Online]. Available: <http://www.accenture.com/NR/rdonlyres/A58111E4-22E5-4DDD-B3DE-FB3741F0052F/0/EmbracingDigitalEco.pdf>
- [45] M. Kulkarni and R. Kreutzer, "Building your own digital ecosystem: a holistic approach to enterprise integration," Syntel, Tech. Rep., 2006. [Online]. Available: <http://www.syntelinc.com/uploadedFiles/Syntel.DigitalEcosystem.pdf>
- [46] M. Vandenberghe, "Digital ecosystem solution - the business factory," XeWOW, Tech. Rep., 2006. [Online]. Available: <http://themaddesigner.free.fr/XeWOW%20White%20Paper.pdf>
- [47] M. Iansiti and R. Levien, *The Keystone Advantage: What the New Dynamics of Business Ecosystems Mean for Strategy, Innovation, and Sustainability*. Harvard Business School Press, 2004.
- [48] F. Nachira, "Towards a network of digital business ecosystems fostering the local development," Directorate General Information Society and Media, European Commission, Tech. Rep., 2002. [Online]. Available: <http://www.digital-ecosystems.org/doc/discussionpaper.pdf>
- [49] M. Papazoglou, "Agent-oriented technology in support of e-business," *Communications of the ACM*, vol. 44, pp. 71–77, 2001.
- [50] K. Soraku-gun, "An evolutionary approach to synthetic biology: Zen and the art of creating life," in *Artificial Life: An Overview*, C. Langton, Ed. MIT Press, 1995, pp. 195–226.
- [51] S. Grand and D. Cliff, "Creatures: Entertainment software agents with artificial life," *Autonomous Agents and Multi-Agent Systems*, vol. 1, pp. 39–57, 1998.
- [52] D. Cliff and S. Grand, "The creatures global digital ecosystem," *Artificial Life*, vol. 5, pp. 77–93, 1999.
- [53] G. Briscoe, M. Chli, and M. Vidal, "Creating a Digital Ecosystem: Service-oriented architectures with distributed evolutionary computing (BOF-0759)," in *JavaOne Conference*. Sun Microsystems, 2006. [Online]. Available: <http://arxiv.org/0712.4159>
- [54] C. Masuch. (2008) Evolutionary environment network. [Online]. Available: <http://evenet.sourceforge.net/>
- [55] T. Kurz. (2008) EvE simulator. [Online]. Available: <http://evesim.sourceforge.net/>
- [56] M. Wooldridge, *Introduction to MultiAgent Systems*. Wiley, 2002.
- [57] S. Wright, "The roles of mutation, inbreeding, crossbreeding and selection in evolution," in *International Congress on Genetics*, D. Jones, Ed. Brooklyn botanic garden, 1932, pp. 356–366.
- [58] D. Hebb, *The Organization of Behavior*. Wiley, 1949.
- [59] D. White and M. Houseman, "The navigability of strong ties: Small worlds, tie strength, and network topology," *Complexity*, vol. 8, pp. 72–81, 2002.
- [60] X. Yang, "Chaos in small-world networks," *Physical Review E*, vol. 63, pp. 1–4, 2001.
- [61] D. Watts and S. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, pp. 440–442, 1998.
- [62] T. Blickle and L. Thiele, "A comparison of selection schemes used in evolutionary algorithms," *Evolutionary Computation*, vol. 4, pp. 361–394, 1996.
- [63] E. Lawrence, *Henderson's dictionary of biological terms*. Pearson Education, 2005.
- [64] T. Soule and J. Foster, "Effects of code growth and parsimony pressure on populations in genetic programming," *Evolutionary Computation*, vol. 6, pp. 293–309, 1998.
- [65] A. Weiss, "Computing in the clouds," *netWorker*, vol. 11, pp. 16–25, ACM Press, 2007.